# Scheduling co-operating stacking cranes with predetermined container sequences *

Dirk Briskorn
Professur für BWL, insbesondere Quantitative Planung
Universität Siegen, Hölderlinstr. 3, D-57068 Siegen, Germany
dirk.briskorn@uni-siegen.de

Panagiotis Angeloudis
Port Operations Research and Technology Centre
Department of Civil and Environmental Engineering, Imperial College London, UK
pa01@ic.ac.uk

Michael G.H. Bell
Port Operations Research and Technology Centre
Department of Civil and Environmental Engineering, Imperial College London, UK
mghbell@ic.ac.uk

November 7th, 2011

## Abstract

Crane scheduling in container terminals is known as a difficult optimization problem that has become even more challenging in recent years with the proliferation of multi-gantry automated stacking cranes. In this paper we present an efficient algorithm solving a subproblem arising in this context, namely deciding the priority of cranes after transportation tasks have been assigned. We tackle this problem for both, twin crane setting and crossover crane setting, and develop graphical models and strongly polynomial algorithms accordingly. A series of experiments is carried out where it is shown that the method can produce optimum solutions within exceptionally small run times.

*Keywords:* Automated Stacking Cranes; Scheduling; Container Terminals; efficient algorithm, shortest path representation.

*Some of the authors would like to express their gratitude to the UK Engineering and Physical Sciences Research Council for financing parts of their work.

# 1 Introduction

Automation has been a key agenda item for port operators and equipment manufacturers over the last 20 years. Seen as the key to significant performance improvements and cost savings, there have been many efforts to automate several terminal operation aspects. While some of these efforts (such as Automated Guided Vehicles - AGVs) have yet to achieve widespread adoption, apart from some early deployments, others types of automation were more successful over the years.

Automated Stacking Cranes (ASCs) represent a major success story from this period and have allowed terminal designers to increase the number of containers processed and stored with less cost and space requirements. Nevertheless, when compared to AGVs, ASCs represent a milder shift from non-automated designs that have been used in the past, specifically Rail Mounted Gantries (RMGs), and as a result they have been perhaps the simplest port unit to automate fully.

Since some or all types of vehicles deployed in the terminal may be unable to lift containers independently it would be essential to fix a time in the planning horizon where both vehicles and gantries would meet at the same place to exchange containers. In practice this requires the presence of a sophisticated scheduling technique that is able to determine a series of crane movements that are coordinated with other terminal activities. While several scheduling algorithms have been developed in the past for this purpose, they often overlook the potential problems that arise from the presence of two or more gantries in the same stack. The study discussed in this paper is part of a greater effort to comprehensively address crane scheduling. The methods developed here can be used to create detailed collision-free movement graphs in advance, that could later be utilized to schedule container moves.

Several problems are considered, each catering to a different combination of crane design and gantry collision avoidance practice. Each problem setting examines a single container block consisting of rows $1, \ldots, R$. As encountered in the design of latest automated container terminals in Europe, the two ends of the block are referred to as rows $0$ and $R + 1$. Each of those two rows will be dedicated to ship- or hinterland-bound moves respectively, and will be the site where containers are exchanged between the cranes and terminal vehicles. These could either be internal vehicles for the transfer of containers between the stack, quay and rail (tugs, straddle carriers, AGVs etc.) or external hauliers belonging to trucks.

For the purposes of this study we assume that the container block is managed by two ASCs which could be either crossover cranes or twin cranes (Figure 1). The crane activities considered will involve containers that are either stored in the block or taken out of the block, with origin and destination positions that are known in advance.

In both configuration cranes cannot move independently but may impede each other. A pair of crossover cranes may pass each other, i. e. both may serve transportation vehicles in row $0$ as well as in row $R + 1$. However, while the larger crane's spreader is releasing or lifting a container in a certain row the smaller crane can neither travel across this row nor perform a release or lift in the same row. In twin configurations the cranes have similar gantry and therefore cannot pass each other. As a result, the two cranes are destined to exclusively serve the opposing ends of the stack, with careful planning required when they need to operate in the middle sections. This also makes it impossible to transfer a container between the two buffers in a single move, which is ultimately not a significant limitation as such moves are practically unheard of in practice.

We assume that both cranes have a sequence of container tasks assigned such that no container assigned appears in both sequences. Furthermore, in the case of twin cranes, each container assigned to the crane serving row $0$ ($R+1$) must be handed over at row $0$ ($R+1$) or must be a container to be relocated within the block. Both cranes must process their respective container tasks in the predetermined sequences. Then, we aim to determine a schedule for both cranes, in the form of sequences of positions over time ensuring that containers are processed in the predetermined order and cranes do not violate the aforementioned movement restrictions. Our goal is to identify the schedule that minimizes the overall length of operations, thst is the crane finishing second does so as early as possible.

This rather narrow problem setting can be motivated easily. First, since the aspect of avoiding collisions is rarely considered in the literature we may take an arbitrary schedule provided by one of the existing approaches and reoptimize the priorities without modifying the sequences of operations assigned to both cranes. Second, of course our algorithm for deciding priorities may be employed in a higher level approach to holistic cranes scheduling. For example, we can easily imagine a metaheuristic approach deciding the sequences of operations assigned to both cranes. In this way, one crane may have to give way to the other (holding a higher priority), as it waits for it to complete its activities in the obstructed area. Clearly, these sequences do not fully specify a schedule which is implementable. There are two options at hand. First, we may extend the representation scheme used in the metaheuristic framework in order to represent priorities, as well. It is likely that we end up with a representation scheme having several sections differing in their semantics. Moreover, redundancy can hardly be avoided. The second option is to design an efficient module deciding about priorities in traffic and to employ it to „interpret“ individuals specified by two sequences of operations. Then, each individual corresponds to many schedules and the module allows us to find the optimum schedule among those represented. This reduces the size of the search space (in comparison to the first option), reduces redundancy, and supports a representation scheme with a uniform structure.

The paper proceeds as follows. In Section 2 we outline related literature and in Section 3 we formally define the problem. Section 4 provides a graphical model representing the problem at

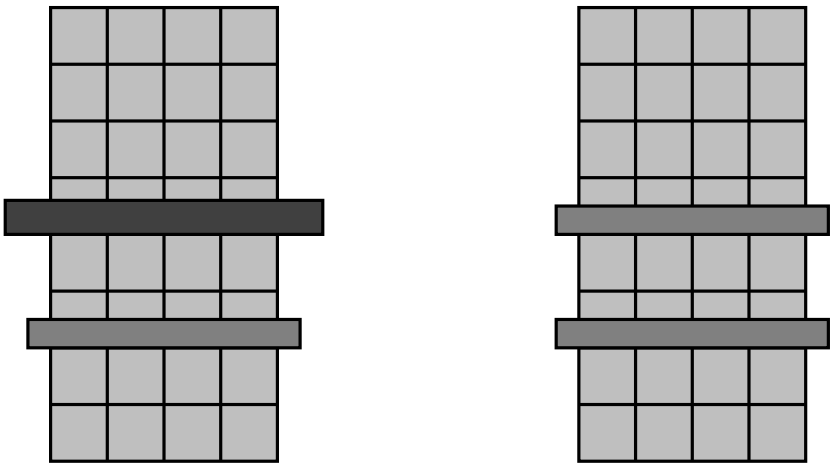

Figure 1: Crossover cranes (left) and twin cranes (right)

hand providing a reduction of our problem to a shortest path problem in a plane with obstacles. The latter problem in turn is reduced to the shortest path problem in a graph and efficiency of the resulting approach is shown in 5. Finally, we conclude the paper in Section 6.

# 2 Previous work

A general overview about operations in a container terminal and corresponding optimization approaches is given in Steenken et al. [9] and updated in Stahlbock and Voß [8]. Vis [10] gives an overview focussing on design and control issues.

The relative novelty of ASCs is reflected in the academic literature, which contains only a few existing studies on this topic. While in the past, several methods were developed for scheduling single (mostly non-automated) stack cranes (Daganzo [2]; Ng and Mak [7]), the presence and interaction between multiple gantries necessitates a novel scheduling technique that determines obstruction-free schedules. Ng [6] was among the first to investigate the optimal scheduling of jobs to multi-gantry cranes, and presents an Integer Programming model that can be used to determine the sequence of crane activities for the execution of a series of container moves. Collisions between gantries are avoided through the enforcement of allowable movement ranges at any point in time. The approach presented in this paper focuses on twin crane configurations, and a dynamic programming-based heuristic is presented accordingly.

A later study by Lee et al. [5] focuses on scheduling of multiple transtainer cranes. These are not common in automated container terminals and contrary to the configurations examined in this study involve a buffer that runs along the length of the stack. Gantry travel while loaded is kept to an absolute minimum, as vehicles are able to approach the gantry from the side.

Vis and Carlo [11] presents a scheduling approach for cross-over ASCs that employs an Integer Programming model which can determine a lower-bound schedule, accompanied by a Simulated Annealing heuristic that can further balance the operations of both gantries. All of the studies mentioned so far make the assumption that crane gantries are solely engaged with simple container-handling, and therefore do not have the ability to directly accommodate reshuffling or other crane activities. This means that the execution of job allocations produced by the aforementioned algorithms will eventually deviate from the algorithm outputs which may lead to difficulties in the accurate synchronization of activities with AGVs and other terminal processes.

A scheduling algorithm for triple cross-over stacking cranes is discussed by Dorndorf and Schneider [3]. As a subproblem the triple crane version of the problem we consider here is tackled. Avoiding collisions for a given sequence of operations is accomplished by employing a branch & bound algorithm. For the instance sizes tested run times are encouraging. However, run times can be expected to increase exponentially in the number of containers considered and the run time complexity of the problem remains open. It should be emphasized that to the best of our knowledge Dorndorf and Schneider [3] provides the only approach so far that allows for an exact solution of the type of problem we are focussing on here.

We contribute to this field by providing an efficient algorithm for two crane problems and prove all considered problems to be solvable in polynomial time.

# 3 Problem definitions

We consider a continuous time horizon where cranes can move to immediately adjacent row positions in a single time unit. We refer to the time interval $[t-1, t]$ with $t \in \mathbb{N}$ as period $t$ in the following. For each container $j$ the durations of container lifts $p_j^l$ and releases $p_j^r$ can be measured in multiples of periods and are assumed to be deterministic and known in advance. For the purposes of this study, these durations include horizontal spreader movements within rows. The origin and destination rows of a container $j$ are denoted as $o_j$ and $d_j$, respectively. Both cranes can travel with the same speed.

A schedule for crane $c$, $c \in \{1, 2\}$, can be specified as a sequence of activities for each period. These can be:

- movement from row $r$ to row $r+1$ with $0 \leq r < R+1$,
- movement from row $r$ to row $r-1$ with $0 < r \leq R+1$,
- lifting a container $j$ in row $o_j$,
- releasing a container $j$ in row $d_j$, or
- waiting in row $r$.

A schedule for a crane $c$ will be deemed feasible if

- after arriving at, waiting in or operating in row $r$ the next activity starts from or takes place in $r$,
- crane $c$ will lift or release a container $j$ only if $j$ has been assigned to $c$,
- a container $j$ is released only after it has been lifted,
- each activity between the lift and release of container $j$ is either waiting or moving, and
- during the lift and the release of container $j$ which takes exactly $p_j^l$ and $p_j^r$ periods, respectively, $c$ is located in $o_j$ and $d_j$, respectively.

The length $l(\sigma^c)$ of a schedule $\sigma^c$ for crane $c$ is equal to the number of periods it covers. A feasible schedule $\sigma$ is a pair $(\sigma^1, \sigma^2)$ where $\sigma^1$ and $\sigma^2$ are feasible schedules for cranes 1 and 2 respectively that do not violate any movement restrictions. These restrictions depend on the specific type of cranes and are detailed in Sections 3.1 to 3.4. The goal of our methodology therefore becomes to find a feasible schedule $\sigma = (\sigma^1, \sigma^2)$ that minimizes the overall makespan $\max\{l(\sigma^1), l(\sigma^2)\}$.

## 3.1 Crossover cranes without detour movements

We assume that cranes 1 and 2 are the larger and smaller cranes respectively that operate on the same block. In this model, these are not allowed to make any detours during their journeys, e. g. in order to give way to the other crane. That is, if two consecutive container lift and release activities take place in rows $r$ and $r'$, then the crane must move in exactly $|r' - r|$ steps, possibly interrupted by waiting, from $r$ to $r'$.

The movement restrictions related to cranes impeding each other for this case are as follows. If crane 1 lifts a container or releases a container in row $r$ during period $t$, then crane 2 cannot

- move from $r$ to $r + 1$, from $r$ to $r - 1$, from $r + 1$ to $r$, or from $r - 1$ to $r$ in $t$,
- wait in $r$ in $t$, or
- lift or release a container in $r$ in $t$.

## 3.2 Crossover cranes with detour movements

In this model the restrictions regarding the simultaneous movements of both cranes are identical to those presented in Section 3.1. Here, however, cranes are free to make a detour in order to give way to the other crane. They may therefore require more than $|r' - r|$ movement steps between two rows $r$ and $r'$ where consecutive container lift and release activities take place.

## 3.3 Twin cranes without detour movements

We assume that cranes 1 and 2 serve rows 0 and $R + 1$, respectively. As in Section 3.1 cranes are not free to make a detour in order to give way to the other crane. The restrictions on simultaneous movements of the cranes require that during each period

- crane 1 stays in row $r$ and crane 2 stays in row $r'$ with $r' \geq r + 1$,
- crane 1 stays in row $r$ and crane 2 moves from $r'$ to $r''$ with $\min\{r', r''\} \geq r + 1$,
- crane 2 stays in row $r$ and crane 1 moves from $r'$ to $r''$ with $\max\{r', r''\} \leq r - 1$, or
- crane 1 moves from $r$ to $r'$ and crane 2 moves from $r''$ to $r'''$ with $r'' \geq r + 1$ and $r''' \geq r' + 1$.

Note that these rules allow crane 1 to move from $r$ to $r + 1$ while crane 2 is moving from $r + 1$ to $r + 2$ in the same period. It would be possible to modify the model to prohibit such simultaneous movements without many complications.

## 3.4 Twin cranes with detour movements

In this model the restrictions on the simultaneous movements of both cranes are identical to those presented in Section 3.3, with the exception that cranes are allowed to make a detours.

# 4 Graphical models

This section outlines a series of graphical models that represent the problems specified in Section 3. These models can provide insights into the problems' structures and allow a reduction to the shortest path problem. Section 4.1 describes an established approach for the graphical representation of the job shop problem with $M$ machines and two jobs. This is later adapted for use with the crane scheduling problems with crossover cranes in Sections 4.2 and 4.3, while the twin crane scenario is tackled in Sections 4.4 and 4.5. We use the following analogy between the job shop problem and the crane scheduling problems:

- Each crane corresponds to a job.
- Each stack row correspond to a job-shop machine.
- Cranes operate in rows in a predetermined order in the same way that jobs visit machines.
- Each crane's operation in a stack row has a given duration just corresponding to the processing time of a job on a machine.

## 4.1 Job shop scheduling with two jobs

In this section we consider two jobs that are processed across $M$ machines in a predetermined sequence. Each job can be processed on each machine an arbitrary number of times, while processing time of each job during each machine visit is assumed to be deterministic and known in advance. Each machine cannot process more than one job at a time. Brucker [1] and Hardgrave and Nemhauser [4] have discussed representations of the job-shop problem using geometric models, with the overall aim being to find the minimum makespan schedule.

The model encompasses a rectangular area with width equal to the total processing time of job 1 and height equal to the total processing time of job 2. Each point $(t_1, t_2)$ in this area represents an overall state where jobs 1 and 2 have completed the first $t_1$ and $t_2$ periods of their respective schedules.

Each pair of visits of jobs 1 and 2 on the same machine induces a rectangular obstacle in the plane. If jobs 1 and 2 are processed on the machine $m$ from $s_1$ to $c_1$ and from $s_2$ to $c_2$, respectively, then we have a rectangular obstacle with corners at $(s_1, s_2)$, $(s_1, c_2)$, $(c_1, s_2)$, and $(c_1, c_2)$. Such obstacles only include the points where the two jobs are being processed in parallel by the same machine, a state that is clearly infeasible. We obtain $\sum_{m=1}^{M}(n_{1,m} \cdot n_{2,m})$ obstacles where $n_{j,m}$ is the number of times a job $j$ is processed on s machine $m$.

Table 1 and Figure 2 illustrate an example for a problem instance and the corresponding model. Table 1 gives the machine index and the corresponding processing time for both jobs, e. g. job 1 is initially processed on machine 1 for 4 periods and then on machine 2 for 2 periods. Job 1 is processed on machine 1 in time intervals $[0, 4]$ and $[22, 24]$ while job 2 is processed on machine 1 in time interval $[0, 2]$. This gives rise to obstacles 1 and 8 in Figure 2.

In terms of the graphical model, the means to find a minimum makespan schedule would be to determine a shortest path from the lower left corner of the area to the upper right corner of the area. A feasible path consists of lines that can only run horizontally, vertically, or diagonally and must not cross through obstacles. A feasible path $p$ depicted in Figure 2 runs

| Job 1 | | Job 2 | |
|---|---|---|---|
| $m$ | $p$ | $m$ | $p$ |
| 1 | 4 | 1 | 2 |
| 2 | 2 | 2 | 6 |
| 3 | 2 | 4 | 3 |
| 4 | 6 | 3 | 2 |
| 5 | 5 | 6 | 2 |
| 6 | 3 | 5 | 3 |
| 1 | 2 | 3 | 2 |

Table 1: Job shop example

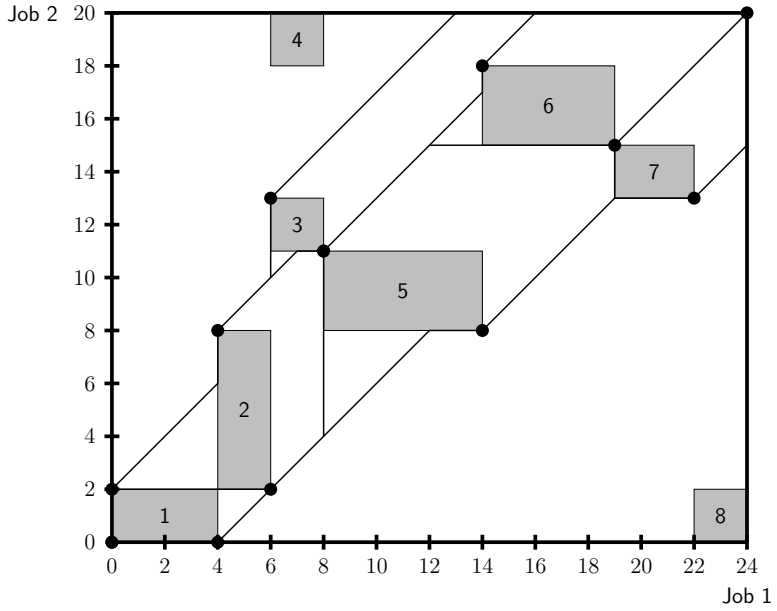


Figure 2: Job shop example

through points $(0,0)$, $(4,0)$, $(12,8)$, $(14,8)$, $(19,13)$, $(22,13)$, $(24,15)$, and $(24,20)$. In the model, horizontal sections of a path correspond to a phase where job 1 is processed and job 2 waits while the opposite applies for vertical sections. For example, the section from $(19,13)$ to $(22,13)$ signifies that job 1 is being processed on machine 6 for 3 periods while job 2 is waiting to be processed on this machine. Diagonal sections suggest that both jobs are being processed.

For the purposes of this study, the lengths of a horizontal segment between $(t^1,t^2)$ and $(t^1+x,t^2)$, a vertical segment between $(t^1,t^2)$ and $(t^1,t^2+x)$, and a diagonal segment between $(t^1,t^2)$ and $(t^1+x,t^2+x)$ are considered to be equal to $x$. Thus, the length of each segment represents the timespan necessary for both cranes to carry out the corresponding activities. Consequently, the length of $p$ in Figure 2 is 29.

It is obvious that there may be an infinite number of paths with the required properties. Thus, reducing this set to a finite set of candidate paths is substantial. Both Brucker [1] and Hardgrave and Nemhauser [4] present techniques for determining finite sets of horizontal, vertical and diagonal sections such that the overall shortest path can be constructed as a concatenation of a subset of these sections. The description of a simple method is taken from Hardgrave and Nemhauser [4].

1. Set $(s^1, s^2) \leftarrow (0, 0)$
2. Starting from $(s^1, s^2)$ move diagonally until an obstacle $o$ is found, say at $(t^1, t^2)$.
3. Branch in two directions.
   (a) If $(t^1, t^2)$ is at the bottom edge of $o$:
      i. Move horizontally along the bottom (left) edge of the rectangle until the lower right corner of $o$ is reached. Go to step 2 using this corner as a new starting point.
      ii. Let $(t'^1, t'^2)$ be the lower left corner of $o$. Move vertically starting at $(t'^1, t^2 - (t^1 - t'^1))$ until the upper left corner of $o$ is reached. Go to step 2 using this corner as a new starting point.
   (b) Else if $(t^1, t^2)$ is at the left edge of $o$:
      i. Move vertically along the left edge of the rectangle until the upper left corner of $o$ is reached. Go to step 2 using this corner as a new starting point.
      ii. Let $(t'^1, t'^2)$ be the lower left corner of $o$. Move horizontally starting at $(t^1 - (t^2 - t'^2), t'^2)$ until the lower right corner of $o$ is reached. Go to step 2 using this corner as a new starting point.
4. If the top or right edge of the outer rectangle is met, move along this edge to the finish point.

The resulting sections for the example discussed above are depicted in Figure 2.

## 4.2 Crossover cranes without detour movements

In this section we develop a graphical model which is semantically identical to the model used for the job shop problem outlined in Section 4.1. Here, a point $(t_1, t_2)$ with $t_1, t_2 \in \mathbb{R}^{\geq 0}$, in the rectangular area represents a state where cranes 1 and 2 have completed their first $t_1$ and $t_2$ time units of their schedule respectively. Note that if the value of $t_c$ is not an integer then the location of crane $c$ may not be clearly defined to be one of the rows of the block under consideration. If crane $c$ moves from row $r$ to $r+1$ in period $t$ then we say that it is located in row $r + t_c - (t-1)$ for each $t - 1 \leq t_c \leq t$ in the following. Similarly, if $c$ moves from row $r$ to $r-1$ during period $t$ then we say that it is located in row $r - (t_c - (t-1))$ for each $t - 1 \leq t_c \leq t$ in the following. This gives an intuitive interpretation of the position of both cranes in each point of time.

We derive non-delay schedules for both cranes from the predetermined sequence of containers assigned to them. The makespan $C_1$ and $C_2$ for the non-delay schedule of crane 1 and 2, respectively, equals to the sum of total operation time and total travel time of crane 1 and 2. In each period in $1, \ldots, C_c$, $c \in \{1, 2\}$, of the non-delay schedule crane $c$ either

1. is located in row $r$, $r = 0, \ldots, R+1$,
2. travels from row $r$, $r = 0, \ldots, R$, to row $r+1$, or
3. travels from row $r$, $r = 1, \ldots, R+1$, to row $r-1$.

Let us consider an example where $R = 6$ and crane 1 starts in row 0, picks up a container in row 2, delivers it to row 4, picks up a second container in row 3, and delivers it to row 2. Both, picking up and releasing, take one period for the first container and take two periods for the second container. Afterwards, crane 1 returns to row 0. Crane 2 starts in row 7, picks up a container in row 2 (one period) and releases it in row 4 (two periods), and, finally, returns to row 7. Figure 3 represents the non-delay schedules by outlining the position of both cranes over time.

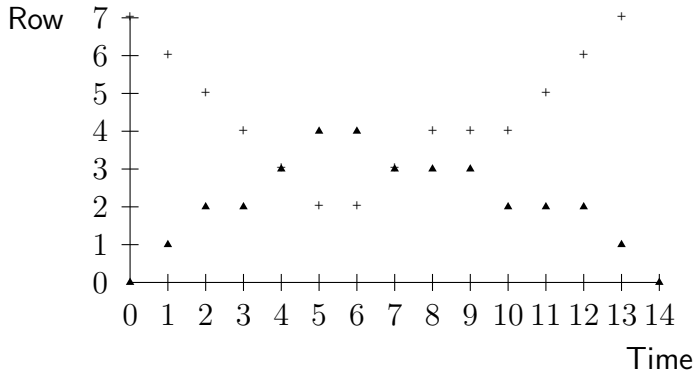


Figure 3: Movement graph depicting the non-delay schedules of crossover cranes

Based on the non-delay schedules we obtain the graphical model depicted in Figure 4. The area's width and the area's height equals $C_1$ and $C_2$. Each point $(t_1, t_2)$ in this area represents an state where cranes 1 and 2 have completed the first $t_1$ and $t_2$ periods of their respective non-delay schedules. An obstacle corresponds to crane 1 operating in row $r$ (and, thus, having its spreader lowered) and crane 2 operating in row $r$, entering $r$, or leaving $r$. The number denoted in each obstacle refers to the corresponding row $r$ where the obstruction potentially takes place.

As in the job shop model we can identify columns and lines of obstacles. Obstacles in the same column correspond to a specific operation of crane 1. Obstacles in the same line correspond to crane 2 moving to a certain row $r$, possibly operating in $r$, and leaving $r$. Figure 4 shows four columns corresponding to crane 1 operating in rows 2, 4, 3, and, again, 2. Furthermore, we have five lines corresponding to crane 2 moving through row 4, moving through row 3, operating in row 2, moving through row 3 again, and operating in row 4. Obstacles corresponding to an operation of crane 2 are divided into three sections. The lower, middle and upper sections correspond to crane 2 moving to row $r$, operating in $r$ and leaving $r$, respectively. Note that the middle section has height zero if crane 2 is moving through the row but not operating in it.

It is not hard to see that the obstacles derived as described above accurately specify the set of infeasible states. Now, the shortest path from $(0, 0)$ to $(C_1, C_2)$ consisting of horizontal sections, vertical sections, and diagonal sections only represents the minimum makespan schedule. However, as can be seen in Figure 4 obstacles may overlap although they are not in the same line or column. Therefore, the method in Hardgrave and Nemhauser [4] cannot be employed as it is. In the following we develop several properties in order to determine a finite set of paths containing the shortest path.

**Property 1.** *Obstacles in different columns do not overlap with respect to their positions regarding the time axis of crane 1.*

Property 1 is due to crane 1 cannot operate in more than one row at a time. However, obstacles in different lines may overlap with respect to their positions regarding the time axis of crane 2 which is different from the job shop model.

**Property 2.** *Obstacles in different lines can overlap only if they correspond to rows $r$ and $r'$ with $|r - r'| = 1$ and if they overlap, then they do by only one period.*

Overlapping is caused by crane 2 occupying both rows when moving from $r$ to $r'$, $|r' - r| = 1$, and crane 1 operating in both, $r$ and $r'$. Now, it is easy to see that Property 2 holds. Also, we can deduce Property 3 from the above.

**Property 3.** *Overlapping of obstacles is line-dependent, that is either each obstacle in line $k$ overlaps with each obstacle in line $k + 1$ or there is no overlapping of obstacles in lines $k$ and $k + 1$ at all.*

We say that an obstacle $o$ is concealed by another obstacle $o'$ with respect to a diagonal if

- $o$ and $o'$ are neither in the same line nor in the same column,
- $o$ and $o'$ overlap with respect to their positions regarding the time axis of crane 2,
- $o$'s position is larger than $o'$'s position with respect to the time axis of crane 2, and
- the diagonal lies left of both, $o$ and $o'$, at the time coordinate of crane 2 where $o$ and $o'$ overlap.

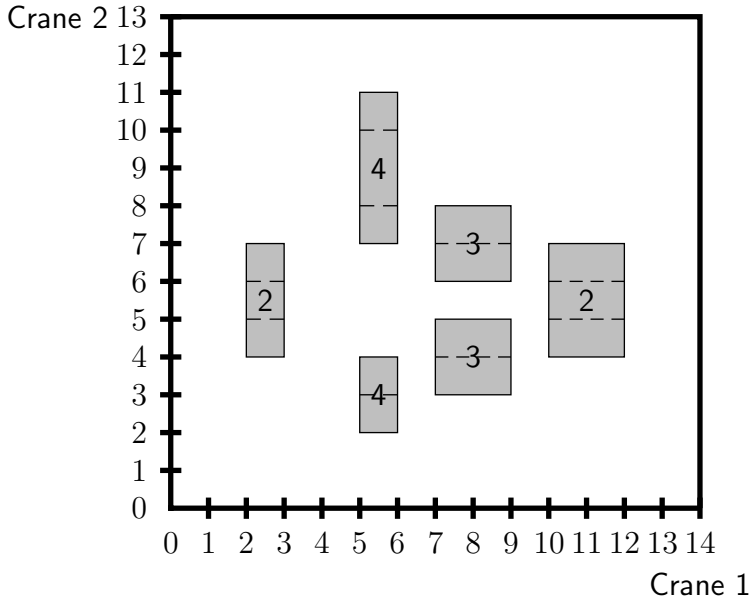


Figure 4: Obstacle graph for a crossover crane configuration

Furthermore, we say that an obstacle $o'$ conceals $o$ indirectly with respect to a diagonal $d$ if there is an obstacle $o''$ such that $o'$ conceals $o''$ (indirectly) with respect to $d$ and $o''$ conceals $o$ (indirectly) with respect to $d$.

Figure 6 illustrates three obstacles resulting from the non-delay schedules given in Figure 5. For the sake of clarity we may say that crane 1 has already picked up a container in row 1 and is about to start to row 5 to deliver it. Afterwards, it picks up a container in row 4, delivers it to row 3, and, finally, moves to row 8. Crane 2 has picked up a container in row 6 and is about to start to row 5 to deliver it. Furthermore, it will pick up a container in row 4, release it in row 3, and move to row 2. Here, obstacle 4 and 5 conceals obstacle 3 and 4, respectively, with respect to the diagonal starting at $(0, 0)$. Furthermore, obstacle 5 conceals obstacle 3 indirectly with respect to this diagonal.

Note that obstacles concealing each other may imply that a path leading below a certain obstacle $o'$ in line $k$ and column $l$ must lead below an other obstacle $o'$ in line $k'$, $k' < k$, and column $l'$, $l' < l$. For example, in the setting according to Figure 6 there is no path leading below 3 but above 4 or 5. The branching process in the approach by Hardgrave and Nemhauser [4] has to be modified accordingly. If a diagonal hits obstacle $o$ we have to construct a branch moving below $o$ which – since we restrict ourselves to schedules without detours – implies that the branch must move below obstacle $o'$, as well, if $o'$ conceals $o$ (indirectly) with respect to the diagonal under consideration. Note that there may be more than one concealing obstacle in line $k - 1$ for an obstacle $o$ in line $k$. However, since all obstacles in line $k - 1$ cover the same periods regarding the time axis of crane 2 we can focus on the concealing obstacle that has largest time coordinates regarding the time axis of crane 1.

The question remains whether it is necessary to move around concealing obstacles. We prove that it is by inspecting the example outlined in Figure 5 and Figure 6. The path depicted in solid lines is the path moving left from the obstacle hit by the diagonal. The path depicted in dashed lines moves below obstacle 3, and since 3 is concealed (indirectly) by 4 as well as 5 with respect to the diagonal starting at $(0, 0)$ it moves below 4 and 5, as well. Note that this path is shorter than the path leading above obstacle 3.

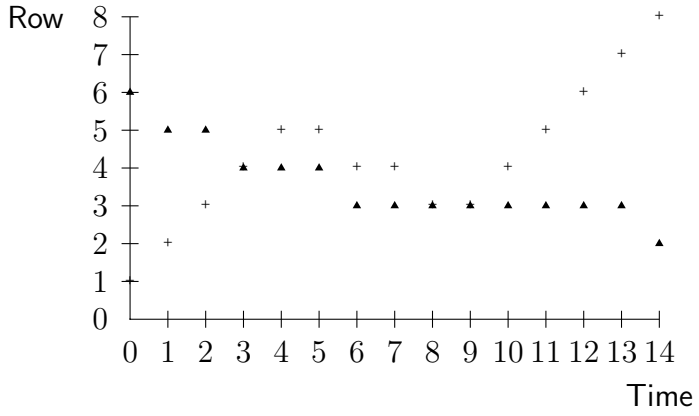


Figure 5: Movement graph for non-delay schedules of crossover cranes

We modify the approach by Hardgrave and Nemhauser [4] in order to find a finite set of paths containing at least one shortest path as follows.

1. Set $(s^1, s^2) \leftarrow (0, 0)$
2. Starting at $(s^1, s^2)$ move diagonally until an obstacle $o$ is reached, say at $(t^1, t^2)$. Let $(t'^1, t'^2)$ be the lower left corner of $o$.
3. Branch into two directions.
   (a) If $(t^1, t^2)$ is at the bottom edge of $o$:
      i. Move horizontally along the bottom edge of the rectangle until the lower right corner of $o$ is reached. Go to step 2 using the corner as a new starting point.
      ii. Move vertically starting at $(t'^1, t^2 - (t^1 - t'^1))$ until the upper left corner of $o$ is reached. Go to step 2 using the corner as a new starting point.
   (b) If $(t^1, t^2)$ is at the left edge of $o$:
      i. Move vertically along the left edge of the rectangle until the upper left corner of $o$ is reached. Go to step 2 using the corner as a new starting point.
      ii. Let $o$ be in line $k$. Find the lowest line $k'$ such that an obstacle in $k'$ conceals $o$ (indirectly). Let $(t''^1, t''^2)$ be the lower left corner of the last concealing obstacle in $k'$. If $t''^2 \geq s^2$, then move horizontally starting at $(t^1 - (t^2 - t''^2), t''^2)$ until the lower right corner of $o'$ is reached and go to step 2 using the corner as a new starting point.
4. If the top or right edge of the outer rectangle is reached then move along that edge to the finish point.

**Theorem 1.** *The set of paths constructed by the modified approach contains at least one shortest path.*

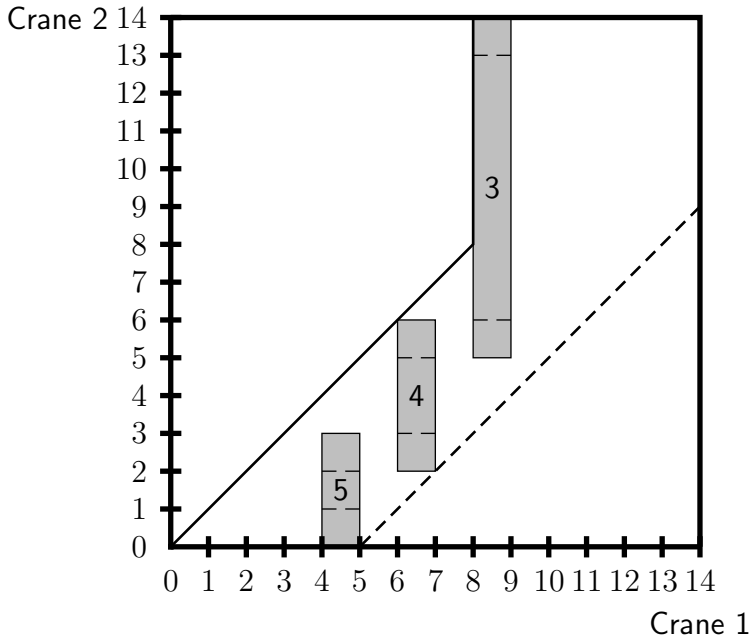


Figure 6: Obstacle graph for the above example

*Proof.* The basic idea of the proof is the same as in Hardgrave and Nemhauser [4] for the original method. It is carried out by induction on the number $h$ of times an obstacle is encountered by a diagonal during construction of the network by our method.

If $h = 0$ (no obstacle is encountered while constructing the network based on a diagonal starting at $d = (d^1, d^2)$), then the modified approach yields a single path only. This path obviously is a shortest path.

Now let us assume that the statement holds for $h-1$. If obstacles are encountered by diagonals $h$ times while constructing the network, consider a network based on a diagonal starting at $d = (d^1, d^2)$. Let $o$ be the obstacle encountered by the diagonal starting at $d$ and let $u$ and $l$ be the upper left corner and the lower right corner used as new starting points for a diagonal. Note that $l$ does not necessarily exist if $(d^1, d^2) \neq (0, 0)$ (see Step 3b of the modified method). However, it does exist if there is any path starting at $d$ and passing below $o$. Note furthermore that $u$ and $l$ do not necessarily belong to the same obstacle. It is obvious that we find the shortest paths from $d$ to $u$ and from $d$ to $l$. Furthermore, we find the shortest paths from $u$ to $C$ and from $l$ to $C$ since obstacles are encountered less than $h$ times in the networks based on diagonals starting at $u$ and $l$, respectively, and, thus, the inductive hypothesis applies. Thus, if there is a shortest path passing through $u$ or passing through $l$, then the network starting at $d$ contains the shortest path from $d$ to $C$.

First, we show that if the shortest path passes below $o$, then we can assume that it passes through $l = (l^1, l^2)$. Consider a shortest path passing below $o$ and not passing through $l$. This path has to go through $(l^1, y)$ for $y < l^2$ and $(x, l^2)$ for $x > l^1$. Note that neither $y > l^2$ nor $x < l^1$ due to the concealing structure of obstacles. Both, the shortest path from $d$ to $l$ and the shortest path from $d$ to $(l^1, y)$, must have length of at least $l^1 - d^1$ since $l^1 - d^1 \geq l^2 - d^2 > x - d^2$. Note that the path from $d$ to $l$ found by our method has this minimum length. Now consider the path from $(l^1, y)$ to $(x, l^2)$ and the horizontal connection from $(l^1, l^2)$ to $(x, l^2)$. Note that the horizontal connection does not cut through any obstacles due to Property 3. Clearly, the shortest path from $(l^1, y)$ to $(x, l^2)$ cannot be shorter as the horizontal connection. Therefore, we can construct a path from $d$ to $(x, l^2)$ through $l$ which is not longer than the connection from $d$ to $(x, l^2)$ on the shortest path.

Next, we show that if the shortest path passes on the left of $o$, then we can assume that it passes through $u = (u^1, u^2)$. The reasoning is analogue to the one above with minor modifications. Note that $u$ belongs to $o$. Here, we consider a shortest path passing left of $o$ and not passing through $(u^1, u^2)$. Then, it must go through $(x, u^2)$ for $x < u^1$ and $(u^1, y)$ for $y > u^2$. We can find another shortest path leading through $u$. Note that the connection from $u$ to $(x, u^2)$ is not concealed according to Property 1.

Summarizing, we find (i) a shortest path from $d$ to $l$ and a network containing the shortest path from $l$ to $C$ according to the inductive hypothesis and (ii) a shortest path from $d$ to $u$ and a network containing the shortest path from $u$ to $C$ according to the inductive hypothesis and we can assume that there is a shortest path from $d$ to $C$ passing through $l$ or through $u$. Thus, our network contains a shortest path if obstacles are encountered $h$ times. This completes the proof. □

## 4.3 Crossover cranes with detour movements

A different approach is required for the case where a crane is allowed to make a detour while traveling between two rows. Note that while a detour increases the total travel distance of the

crane under consideration, it may allow us to determine a schedule with a shorter makespan. To show this, we revisit the example provided in Figure 5.

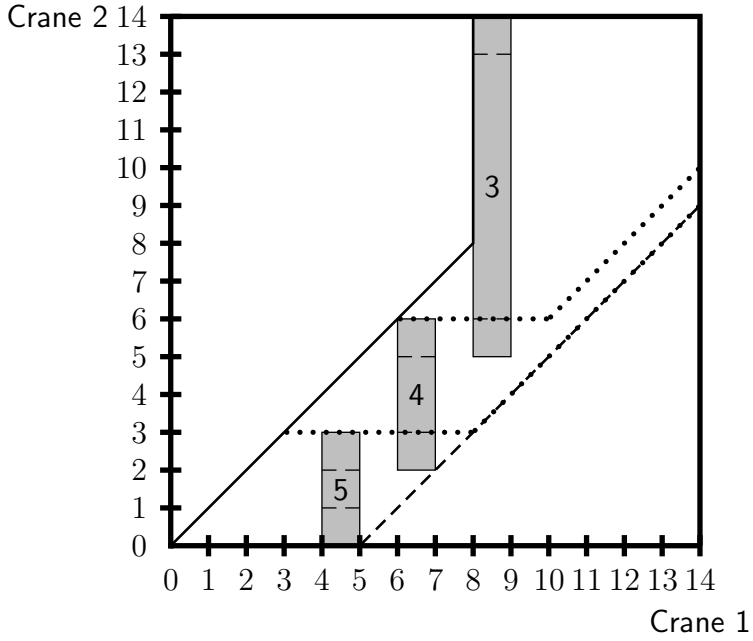


Figure 7: Obstacle graph for crossover crane with detours

In addition to the two paths considered in Section 4.2, Figure 7 includes a third path $p$ leading from $(0,0)$ through $(3,3)$, $(8,3)$, and $(14,9)$ to $(C_1,C_2)$ and a fourth path $p'$ leading from $(0,0)$ through $(6,6)$, $(10,6)$, and $(14,10)$ to $(C_1,C_2)$. Path $p$ corresponds to Cranes 1 and 2 simultaneously moving to row 4 in the first 3 periods. Afterwards, Crane 2 does not start its operation in row 4 but rather makes a detour that allows Crane 1 to operate in row 4 first. Crane 2 returns to row 4 as soon as Crane 1 has finished its operation in row 4. Then, both cranes proceed according to their schedules in parallel. Crane 1 finishes after 14 periods. At this point of time, Crane 2 still has 5 unprocessed periods of its schedule. Since Crane 2 is not delayed anymore it concludes its operations after 19 periods. According to path $p'$ Cranes 1 and 2 simultaneously process their respective schedules for the first 6 periods. Then, Crane 2 makes a detour in order to let Crane 1 operate in row 3 first. Crane 2 returns to row 34 as soon as Crane 1 has finished its operation in row 3. Afterwards, both process their schedules in parallel, Crane 1 finishes after 14 periods and Crane 2 finishes after 18 periods. Thus, $p'$ is shorter than the paths considered in Section 4.2 which proves that detours may enable us to shorten the makespan.

Note that reresenting a detour by a dashed line connecting states $s$ and $t$ does not imply that states in between $s$ and $t$ are actually reached. We only know that both, $s$ and $t$, are reached and that the timespan covered by the detour equals distance of $s$ and $t$. Note also that while doing a detour a crane does not proceed along its non-delay schedule and, thus, detours correspond to either horizontal or vertical segments.

In the following we restrict the type of detours we have to consider without losing optimality of the resulting schedule.

**Property 4.** *There is an optimum schedule such that*

1. *cranes deviate from their prescribed row sequence only in order to reduce waiting time of the other crane,*

2. *crane 1 does not deviate from its row sequence,*

3. *crane 2 starts a detour only in a row where both cranes are going to operate next,*

4. *crane 2 returns to row $r$ where it started its detour as soon as possible, that is, it starts entering $r$ right after crane 1 finished an operation in $r$, and*

5. *crane 2 starts a detour in row $r$ only if both cranes approach $r$ from the same row (either $r-1$ or $r+1$), and Crane 1 has operated in $r-1$ since Crane 2 left $r-1$.*

The first part of the property follows from the fact that a crane detour increases its total travel distance and therefore its makespan. Consequently, we consider a detour only if it implies a reduction on the makespan of the other crane. Since neither travel times nor operation times can be influenced we can only reduce the makespan by reducing the total waiting time. Then, as stated by the second part of the property, crane 1 never has to make a detour since it can avoid delaying crane 2 by simply not operating. Third, crane 2 may delay crane 1 only in a row where crane 1 operates since both cranes do interfere only if crane 1's spreader is lowered. Furthermore, a detour is necessary only if Crane 2 is about to operate in the same row since otherwise it can move out of Crane 1's way following its prescribed row sequence. Fourth, Crane 2 has to return to the row $r$ where it started its detour, naturally, since $r$ is its next operation's row due to the third part of this property. Then, Crane 2 may reach $r$ one period after Crane 1 has finished operating in $r$ since Crane 2 may wait in a neighboring row. The fifth part of the property can be justified as follows. Assume that Cranes 1 and 2 approach $r$ from different rows. Then, Crane 2 can simply wait in the neighboring row of $r$ it passes until Crane 1 finished its operation in row $r$. Crane 2 would reach $r$ one period after Crane 1 finished operating in $r$ (which is as early as possible) without making a detour. Assume that cranes 1 and 2 approach $r$ from the same row, let us say $r-1$ w. l. o. g., but Crane 1 did not operate in $r-1$ before. Then, Crane 2 can simply wait in $r-1$ until Crane 1 finished its operation in row $r$. Again, Crane 2 would reach $r$ not later than before without making a detour. Note that the fifth property exactly specifies two overlapping obstacles of different lines.

In terms of our graphical model this means we can restrict ourselves to detours that

- start at a diagonal $d$ which encounters an obstacle $o$ in line $k$ (according to the first part of Property 4),
- run horizontally (according to the second part of Property 4),
- cut either through $o$ or through obstacle $o'$ in line $k'$, $k' \leq k$, concealing $o$ (indirectly) with respect to $d$ if $o'$ is concealed with respect to $d$ itself (according to the fifth part of Property 4),
- cut through an obstacle $o'$ one period above its lower boundary (according to the third part of Property 4), and
- end one period right of a obstacle in line $k'$ concealing $o$ (according to the fourth part of Property 4).

Next, we apply two more restrictions.

**Lemma 1.** *Assume that a diagonal $d$ encounters an obstacle $o$ in line $k$. There is an optimum schedule such that if a detour starts at $d$ and cuts through obstacles in line $k'$, $k' < k$ , then it will cut through each obstacle in line $k$ concealing $o$ with respect to $d$.*

*Proof.* Consider an optimum schedule represented by a path $p$ and a detour being part of $p$ and starting at $(s^1, s^2)$ from diagonal $d$ which encounters obstacle $o$. Assume that this detour cuts through at least one obstacle in line $k'$ but it does not cut through all obstacles in $k'$ concealing $o$ with respect to $d$. Furthermore, assume that this detour is the first detour of this kind in $p$. Clearly, either (i) $p$ must lead to the upper left corner $u$ of $o$ or (ii) at least one more detour cutting obstacles concealing $o$ with respect to $d$ is used in $p$.

(i) We can find a path $p'$ which is identical to $p$ except for the connection from $(s^1, s^2)$ to $u$. We can simply follow $d$ until it encounters $o$ and, then, go vertically to $u$. This connection cannot be longer than the one in $p$ and, thus, $p'$ cannot be longer than $p$. Note that the number of detours not cutting all concealing obstacles in a line in $p'$ is smaller than in $p$.

(i) Let $(t^1, t^2)$ be the starting point of the detour being used next. We can find a path $p'$ which is identical to $p$ except for the connection from $(s^1, s^2)$ to $(t^1, t^2)$. We can follow $d$ until $(s^1 + (t^2 - s^2), t^2)$ and start a detour. This detour will lead through $(t^1, t^2)$. Note that $p'$ cannot be longer than $p$ and the number of detours not cutting all concealing obstacles in a line in $p'$ is smaller than in $p$.

By repeating the step described above we can find a schedule as described in Lemma 1. This completes the proof. □

**Lemma 2.** *There is an optimum schedule where a detour starting from diagonal $d$ cuts only obstacle $o$ encountered by $d$.*

*Proof.* According to Lemma 1, each path containing a detour starting at a diagonal $d$ encountering obstacle $o$ must either pass $o$ below or cuts $o$. We shall show that the shortest path using a detour starting at $(s^1, s^2)$ and passing $o$ below cannot be shorter than the shortest path using a detour starting at $d$ and cutting through $o$.

Consider the point $(t^1, t^2)$ where a detour cutting through $o$ ends. Clearly, each path passing below $o$ must pass through $(t^1, t'^2)$, $t'^2 \leq t^2$, since $(t^1 - 1, t^2 - 1)$ is the lower right corner of $o$. Note that both connections, from $(s^1, s^2)$ to $(t^1, t'^2)$ and from $(s^1, s^2)$ to $(t^1, t^2)$, have length of $t^1 - s^1$. Now, with similar arguments as before it is easy to see that reaching $(t^1, t^2)$ is dominant to reaching $(t^1, t'^2)$ at the same point of time (distance to $(0, 0)$) since the schedule of Crane 2 has been processed further. This completes the proof. □

We outline Lemma 2 using Figure 7, again. The upper detour dominates over the lower one since $(10, 6)$ is reached after 10 periods with the upper detour as $(10, 5)$ is reached after 10 periods following the lower detour.

Lemma 2 provides insights that enable us to modify the method for network generation proposed in Section 4.2 for scenarios with detours. We therefore introduce the following step to the approach outlined at the end of Section 4.2.

3. (b) iii. If $o$ is concealed with respect to the the diagonal reaching $o$, then move horizontally from $(t^1 - (t^2 - (l^2 + 1)), l^2 + 1)$ to $(l^1 + 1, l^2 + 1)$ where $(l^1, l^2)$ is the lower right corner of $o$.

## 4.4 Twin cranes without detour movements

In this section we develop a graphical model for scenarios with twin cranes that are not allowed to make detours. The idea is pretty much the same as in previous sections. Note that in line with the interpretation of state $(t_1, t_2)$ outlined in Section 4.2 we can say that Cranes 1 and 2 are acceptably positioned in rows $r_1$ and $r_2$ if at any time $r_1 \leq r_2 - 1$. This case however involves a new type of obstacle that describes conflicts of the operations or moves of both cranes.

Clearly, Crane 1 operating in $r$ is in conflict to Crane 2 operating in $r'$ if $r > r' - 1$. In the following we consider an obstacle corresponding to each pair of operations of Cranes 1 and 2 being in conflict. Such an obstacle covers the time coordinates where both operations are in conflict, those where the operation of one crane is in conflict with movements of the other crane related to its operation, and those where movements of both cranes related to these operations are in conflict with each other. In the following we specify such an obstacle under the assumption that both cranes enter the area of conflict as late as possible before starting the operations and both cranes leave this area as fast as possible after completing the operation. Note that this assumption is not necessarily justified by the non-delay schedules.

We now formally specify the obstacle corresponding to Cranes 1 and 2 operating in rows $r_1$ and $r_2$, $r_1 \geq r_2$, in periods $t_1^1, \ldots, t_1^2$ and $t_2^1, \ldots, t_2^2$, respectively. We consider an area $C$ that corresponds to the operations themselves neglecting movements. Area $C$ is well defined as a rectangle with a corner at $(t_1^1, t_2^1)$, $(t_1^2, t_2^1)$, $(t_1^1, t_2^2)$, and $(t_1^2, t_2^2)$. The obstacle consists of an area $C$ and a circlet containing states which do not belong to $C$ and have a shortest Euclidian distance to a state in $C$ of less than $r_1 - r_2 + 1$.

In order to provide an intuitive understanding we consider an example where Crane 1 operates in row 8 in periods 9 and 10, while Crane 2 operates in row 5 in periods 7, 8 and 9. We assume here that Cranes 1 and 2 start in rows 0 and 11 (respectively) at time 0 and return there afterwards. Hence, given the rows and times of operation the obstacle will be as small as possible. The corresponding obstacle is depicted in Figure 8.

Each obstacle is composed by nine parts. The dark-grey shaded area labeled $C$ covers the time coordinates where both cranes are operating. The remaining parts correspond to movements being in conflict with operations or other movements and form the circlet mentioned above. The light-grey shaded areas correspond to one of both cranes operating while the other one is moving. Here, $B$ corresponds to Crane 2 passing the row Crane 1 is operating in and approaching the row where it operates next while $T$ corresponds to Crane 2 passing the row Crane 1 is operating in after the leaving the row of its own operation. Accordingly, $L$ and $R$ can be interpreted by switching Cranes 1 and 2 in the description above. The specification of the light-grey shaded areas can be easily derived from the above. Heights of $L$ and $R$ and widths of $T$ and $B$ correspond to the duration of operations Crane 2 and Crane 1, respectively. Widths of $L$ and $R$ and heights of $T$ and $B$ equals the duration of travel in the conflicting area, that is $r_1 - r_2 + 1$. Finally, white areas correspond to movements of Crane 1 being in conflict with movements of Crane 2. The area $TL$ corresponds to Crane 2 moving towards row $R + 1$ after operating and Crane 1 moving towards row 8 in order to operate. Since we

assume that Cranes 1 and 2 can simultaneously move from $r$ to $r+1$ and from $r+1$ to $r+2$, respectively, we obtain the straight line from $(4, 9)$ to $(8, 13)$. We can reason similarly for $BR$, $BL$, and $TR$.

From the description above it is clear that each time coordinate covered by an obstacle corresponds indeed to infeasible positions of Cranes 1 and 2. The reasoning can be stated as follows. Starting from infeasible positions of operations $r_1$ and $r_2$ total movement of both cranes must bridge $r_1 - r_2 + 1$ rows in order to reach positions $r'_1$ and $r'_2$ with $r'_1 \leq r'_2 - 1$. We will now show that, moreover, there is no infeasible time coordinate of Cranes 1 and 2 not covered by an obstacle corresponding to a pair of conflicting operations of cranes 1 and 2.

**Lemma 3.** *Each infeasible time coordinate of Cranes 1 and 2 is covered by at least one obstacle corresponding to a pair of conflicting operations of cranes 1 and 2.*

*Proof.* We distinguish between infeasible states, i. e. time coordinates, corresponding to (i) both cranes operating, (ii) one crane operating and the other crane moving, and (iii) both cranes moving. Consider a specific infeasible time coordinate $(t^1, t^2)$.

(i) Clearly, each infeasible time coordinate is covered by area $C$ of an obstacle.

(ii) We restrict ourselves to crane 1 operating and crane 2 moving. For the other case the reasoning is analogue.

Let us assume that crane 1 is operating in $r^1$ at time $t^1$ of its non-delay schedule and crane 2 is moving from the last operation's row $r_1^2$ to the next operation's row $r_2^2$ at time $t^2$ of its non-delay schedule. First, if $r_1^2 < r_2^2$, then $r_1^2 < r^1$ and crane 2 is strictly moving from row $r_1^2$ towards row $R+1$ at time $t^2$. Thus, $(t^1, t^2)$ must be covered by the obstacle corresponding to the operations of crane 1 in $r^1$ and crane 2 in $r_1^2$. Second, if $r_1^2 > r_2^2$, then $r_2^2 < r^1$ and crane 2 is strictly moving towards row $r_2^2$ at time $t^2$. Thus,

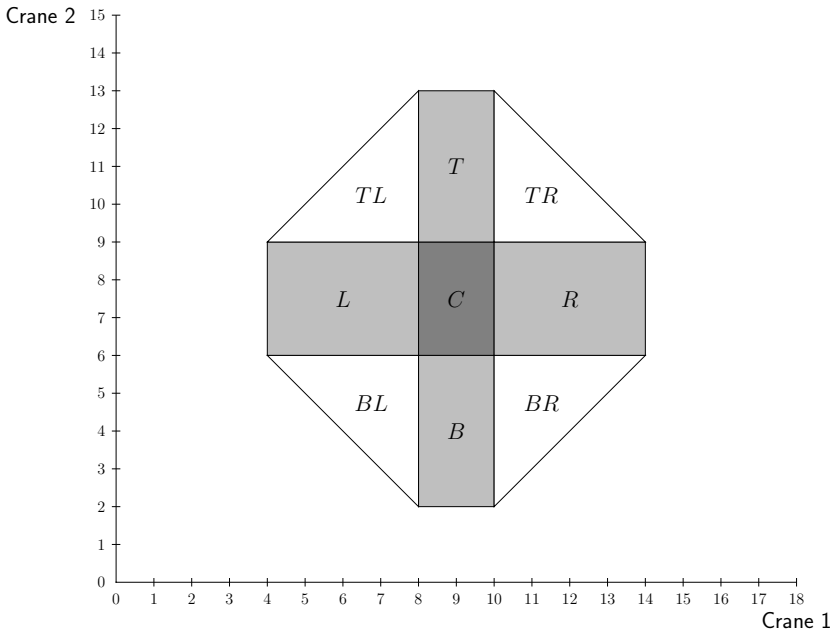


Figure 8: Obstacle graph for twin cranes

$(t^1, t^2)$ must be covered by the obstacle corresponding to the operations of crane 1 in $r^1$ and crane 2 in $r_2^2$.

(iii) Let crane $c$ be moving from the last operation's row $r_1^c$ to the next operation's row $r_2^c$ in $(t^1, t^2)$. In each of the following cases a similar reasoning as in (ii) applies.

- If $r_1^1 < r_2^1$ and $r_1^2 < r_2^2$, then $(t^1, t^2)$ must be covered by the obstacle corresponding to the operations of crane 1 in $r_2^1$ and crane 2 in $r_1^2$.
- If $r_1^1 < r_2^1$ and $r_1^2 > r_2^2$, then $(t^1, t^2)$ must be covered by the obstacle corresponding to the operations of crane 1 in $r_2^1$ and crane 2 in $r_2^2$.
- If $r_1^1 > r_2^1$ and $r_1^2 < r_2^2$, then $(t^1, t^2)$ must be covered by the obstacle corresponding to the operations of crane 1 in $r_1^1$ and crane 2 in $r_1^2$.
- If $r_1^1 > r_2^1$ and $r_1^2 > r_2^2$, then $(t^1, t^2)$ must be covered by the obstacle corresponding to the operations of crane 1 in $r_1^1$ and crane 2 in $r_2^2$.

This completes the proof. □

At this point we can specify the set of infeasible states as the union of interior points of obstacles corresponding to pairs of conflicting operations. We outline an example in Figure 9 and Figure 10.

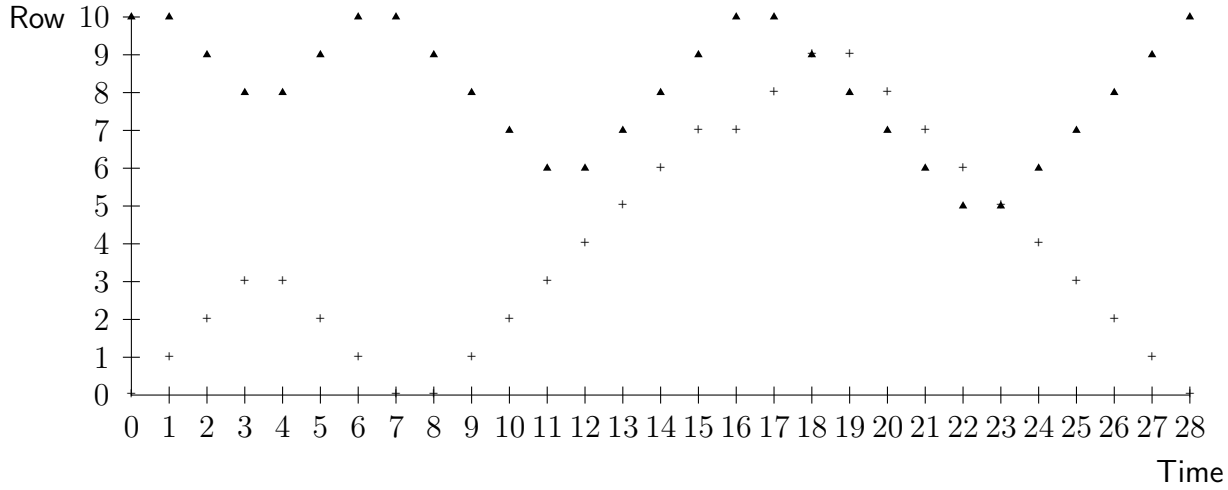


Figure 9: Movement graph for non-delay schedule of twin cranes

We have five obstacles in Figure 10 corresponding to pairs of conflicting operations of cranes 1 and 2 that are carried out in rows 7 and 6, 7 and 5, 9 and 8, 9 and 6, and 9 and 5. As it can be seen the obstacles here may overlap. Also, obstacles in the same line (or column) may differ with respect to their height or width. Strictly following the same idea as in Sections 4.2 and 4.3, we encounter the following problem: Assuming the diagonal reaches an obstacle $o$, we have to design a path passing $o$ below. The problem in this occasion is to find the highest line of obstacles such that at least one obstacle in this line is connected to $o$ by a sequence of pairwise overlapping obstacles. An analogue problem occurs when designing the path passing $o$ at the left.

Instead, we propose to design a path passing through the bottom border of each obstacle on the right of the diagonal that is not concealed with respect to the diagonal whose bottom border is higher than the diagonal's starting point. The term concealed in this case suggests that the horizontal connection from the diagonal to the obstacle would lead through an other obstacle. In the example in Figure 10 we can see that the diagonal starting in $(0, 0)$ is connected to the bottom border of four obstacles. Consequently, a path is established passing through the left border of each obstacle above the diagonal and not concealed with respect the diagonal whose left border is right of the starting point of the diagonal. Note that the diagonal starting in $(0, 0)$ is connected to one obstacle only since the diagonal passes above three obstacle and one the remaining obstacles is concealed with respect to the diagonal.

What remains to be shown is that it is sufficient to consider horizontal connections passing through the bottom borders of obstacles and vertical connections passing through left borders only. This can be shown easily by applying reasoning similar to the ones applied in Section 4.2. For the sake of shortness we do not go into detail here.

## 4.5 Twin cranes with detour movements

In this section we modify the graphical model developed in Section 4.4 in order to cover crane schedules including detours. First, we shall outline that indeed detours may improve schedules in terms of makespan minimization. We outline an example in Figure 11 and Figure 12.

The strategy outlined in Section 4.4 provides two options: The path moving below both overlapping obstacles has a length of 21 and the path that leads to the left of the obstacles has a length 22. The third path depicted in Figure 12 corresponds to Crane 1 operating in row 2 first, then dodging Crane 2 that operates in row 1, returning to row 2 as soon as Crane

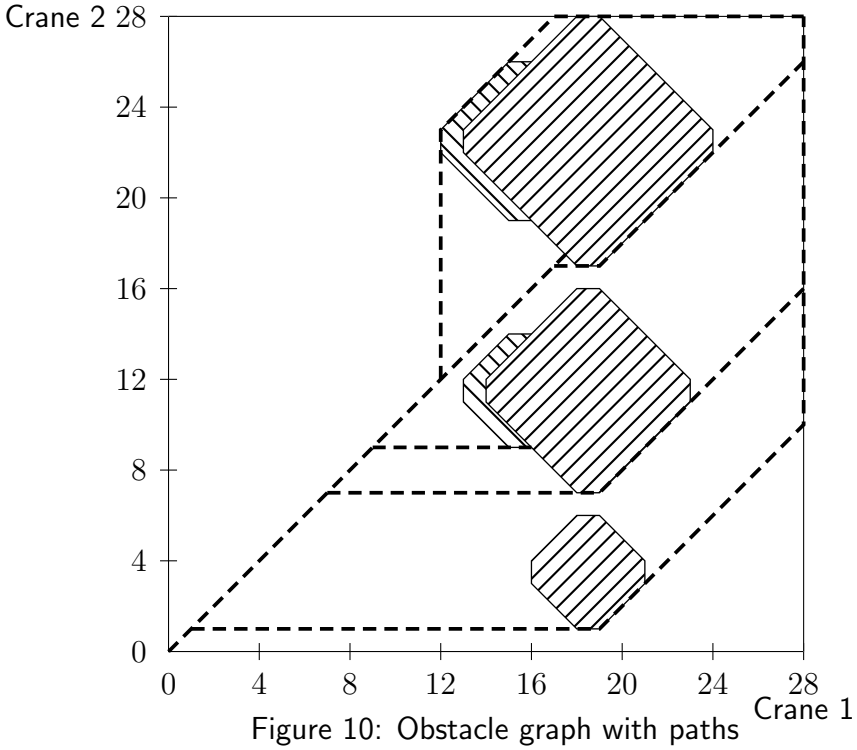

Figure 10: Obstacle graph with paths

2 leaves, and, finally, moving to row 5 and returning to row 0. This path has length 20 and, thus, is the shortest path among these three.

In the following we state several properties of optimum schedules. We do so for Crane 1 being the crane making a detour. Properties for Crane 2 making a detour are strictly analog.

**Property 5.** *There is an optimum schedule such that*

1. *Crane 1 makes at most one detour between each pair of consecutive operations.*

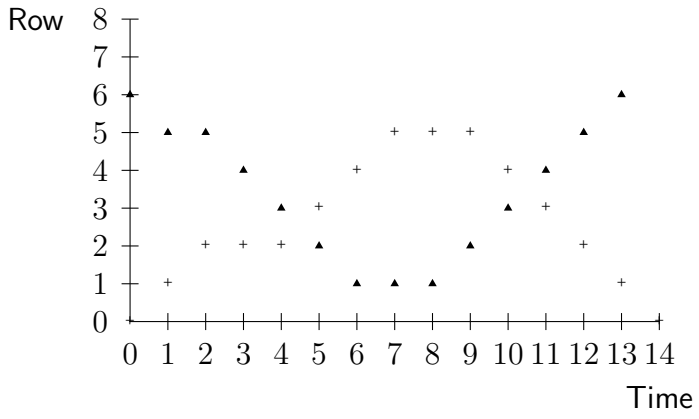


Figure 11: Movement graph for a twin crane non-delay schedule

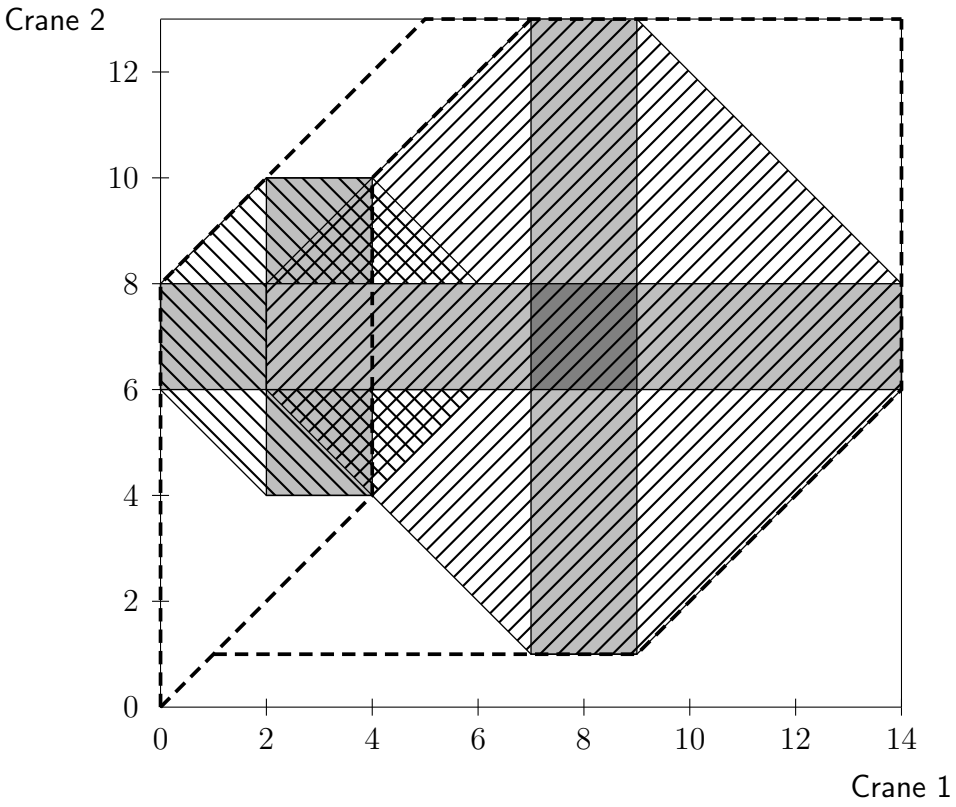


Figure 12: Obstacles graph with paths for the above example

2. *Crane 1 starts a detour between a pair of consecutive operations in rows $r$ and $r'$*
   (a) *immediately before starting the second operation if $r' \leq r$ or*
   (b) *immediately after completing the first operation if $r \leq r'$.*
3. *Crane 1 returns to row $r$ it started the detour in immediatly after crane 2 completes an operation in row $r$.*

The first part of the property is deduced as follows: Assuming that there is more than one detour of Crane 1 after completing $o$ and before starting $o'$. Let $r$ be the row among those travelled on these detours that has largest distance to the rows where both operations take place. It can be seen easily that a single detour to $r$ suffices in order to give way to crane 2.

The second part of the property can be justified as follows. We can restrict ourselves to detours starting either immediately after an operations has been completed or immediately before an operations has been started since otherwise the crane runs into one direction and turns around at some point. Now, first, assume that Crane 1 starts a detour immediately after completing the first operation $r' \leq r$. Clearly, it can move to row $r'$ before starting its detour since moving from $r$ to $r'$ means giving way to crane 2. Second, assume that Crane 1 starts a detour immediately before starting the second operation if $r < r'$. Then, the crane runs into one direction and turns around at some point which as mentioned before is not efficient. Instead it could either wait in $r$ and not make a detour at all if it does not impede crane 2 in row $r$ or start its detour from row $r$.

The third part of the property is easy to see since we assume that both cranes move with same speed. Thus, in the same period crane 2 leaves the row where crane 1 started its detour crane 1 can return to this row.

We can represent detours according to Property 5 by an extension of the graphical model developed in Section 4.5.

- If a path passes below the lower right corner of part $B$ of an obstacle corresponding to Crane 1 operating in $r$ and the next operation of Crane 1 takes place in row $r'$, $r' \geq r$, we connect this path vertically to the upper right corner of $T$ of this obstacle.
- If a path passes below the lower left corner of part $B$ of an obstacle corresponding to Crane 1 operating in $r$ and the previous operation of Crane 1 takes place in row $r'$, $r' \geq r$, we connect this path vertically to the upper left corner of $T$ of this obstacle.
- If a path passes left of the upper left corner of part $L$ of an obstacle corresponding to Crane 2 operating in $r$ and the next operation of Crane 2 takes place in row $r'$, $r' \leq r$, we connect this path horizontally to the upper right corner of $R$ of this obstacle.
- If a path passes left of the lower left corner of part $L$ of an obstacle corresponding to Crane 2 operating in $r$ and the previous operation of Crane 2 takes place in row $r'$, $r' \leq r$, we connect this path horizontally to the lower right corner of $R$ of this obstacle.

Note that in the instructions above only paths as constructed in Section 4.5 are considered, that is we do not connect a section corresponding to a detour of Crane 1 with a section corresponding to a detour of Crane 2.

# 5 Algorithm Implementation and Computational Results

Following the basic idea provided in Brucker [1] we can represent the problem to find the shortest path in the networks constructed in Sections 4.2 to 4.5 as the problem to find a shortest path in a directed acyclic graph (DAG). We first give a generic description of the graph corresponding to each of the graphical models and discuss complexity issues afterwards.

We consider a weighted directed graph $G = (V, E, w)$ where

- $V = \{i \mid i$ is a starting point of a diagonal or $(C_1, C_2)\})$
- $E = \{(i, j) \mid i \in V, j \in V, i$ is directly connected to $j$ in the network$\}$
- $w_e = w_e^{di} + w_e^{hv} + w_e^{de}$ where $w_e^{di}$, $w_e^{hv}$, and $w_e^{de}$ is the length of the diagonal section, the length of the horizontal/vertical section, and the length of the detour section which $e$ consists of (note that each of the lengths may equal zero)

First, we see that $G$ is acyclic since $j^1 > i^1$ or $j^2 > i^2$ (possibly both) holds for each $e = ((i^1, i^2), (j^1, j^2)) \in E$. Thus, the shortest path in $G$ can be found in $O(|E|)$ time. Clearly, we have $O(n^2)$ obstacles in each model. There are three or two starting points of diagonals associated to each obstacle depending on whether detours are allowed or not. Thus, we have $|V| \in O(n^2)$. In the crossover crane models each starting point of a diagonal section is connected to at most three or two other starting points depending on whether detours are allowed or not. Thus, in the crossover case we end up with $|E| \in O(n^2)$. In the twin crane case, however, each starting point of a diagonal section is potentially connected to $\Omega(n^2)$ other starting points even if detours are not allowed. Clearly, if detours are allowed no point is connected to more than $O(n^2)$ other points. Hence, in the crossover case we end up with $|E| \in O(n^4)$. Note that, remarkably, the run time complexity does not depend on $R$.

The graph models have been therefore shown to provide strongly polynomial solution methods for the crane scheduling problems considered. To test the algorithm, we developed a software tool called CraneGraphs using C#, that translates job sequences into a schedule of movements. The tool proceeds as follows for a given pair of sequences of transport jobs:

- Using the sequences of jobs, construct a non-delay schedules for both cranes.
- Potential collisions between the two gantries are identified in accordance to the crane configuration that is in place.
- Create graph $G$ for the crane configuration under consideration.
- Use Dynamic Programming to find a shortest path in $G$ (recall that $G$ is acyclic).
- Edges in the shortest path are translated into the corresponding segments in the graphical model which are in turn translated into a set of detours and movement delays that are injected into the original non-delay schedule.

On a moderately modern desktop computer (with a quad core, 3 GHz Intel Core 2 X9650 CPU and 8 GB of RAM) the algorithm implementation outlined above was capable of delivering results in periods significantly smaller than 1ms. A series of computational experiments were

| Scenario | Twin | Crossover |
|---|---|---|
| 5 (a) | 0.03557ms | 0.03557ms |
| 5 (b) | 0.04168ms | 0.04928ms |
| 10 (a) | 0.07997ms | 0.08130ms |
| 10 (b) | 0.10248ms | 0.10881ms |
| 20 (a) | 0.24132ms | 0.24809ms |
| 20 (b) | 0.34790ms | 0.35421ms |

Table 2: Scenario running times

carried out considering both cross-over and twin configurations. Various types of scenarios were used to assess performance, which varied in terms of crane configuration (twin or crossover), amount of jobs allocated per crane (5, 10 or 20) and the likely amount of overlap (and therefore obstructions) during the operation of the two gantries. For the latter two cases were considered, one with less (a) and another with frequent (b) overlap. The gantries were assumed to be operating on a stack with 30 rows. Several problem instances were created (1,000 for every combination of the above parameters), using different random seeds in the generation process. In order to obtain reliable measurements on the computational times required by the algorithm, each different problem was executed 1,000 times (therefore a total of 1,000,000 executions per each scenario type).

Table 2 outlines average running times for the test cases. What we can see clearly, here, is that running times are exceptionally small and already suitable for making crane scheduling decisions in real-time. We, furthermore, see that running times for crossover instances are lower than those for twin instances which is in line with our worst-case analysis provided above. Also, instances with high interference require higher running times than those with low interference which can be explained by the higher number of obstacles in the twin crane case and the higher number of obstacles being encountered by the network in the crossover case. The implementation used for the purposes of this study did not utilize the presence of multiple CPU cores in the test machine. While this presents an opportunity for further improvement, we believe that the tool as it exists is fast enough for real-world applications.

# 6 Summary and future work

Automated multi-gantry cranes are currently enjoying increasing adoption rates in modern container terminals, and the current outlook is that more terminal operators will implement such technologies in the years to come. While in practice each of the gantries will be focusing in different sides of the same stack interactions in the middle are common that are difficult to acknowledge in job allocation algorithms. This paper has presented a novel gantry scheduling algorithm that can determine optimal, precise and collision-free movement sets in real-time. The algorithm can accommodate any crane operation that can be represented in a movement graph, including container transfers to/from terminal vehicles and container reshuffling. Given its flexibility and sort execution times the approach is receptive for implementation in practice.

There are two main directions for future research. First, future work may seek to embed this technique in job allocation algorithms that sequence container moves and co-ordinate crane activities with other terminal operations. Given the high-levels of accuracy and the fact that

movement schedules can include any type of crane activity (not only container moves), the technique could also be used to develop scheduling methods that better synchronize AGV and ASC operations. Second, future work may seek to generalize the approach to different crane setting, such as triple cranes or cranes with handover positions on the long side of the block, or to different objectives, such as minimization of lateness penalties for containers having delivery due dates.

# References


[1] P. Brucker. An efficient algorithm for the job-shop problem with two jobs. *European Journal of Operational Research*, 40(4):353–359, 1988.

[2] C. F. Daganzo. The crane scheduling problem. *Transportation Research B*, 23:159–175, 1989.

[3] U. Dorndorf and F. Schneider. Scheduling automated triple cross-over stacking cranes in a container yard. *OR Spectrum*, 32:617–632, 2010.

[4] W. W. Hardgrave and G. L. Nemhauser. A geometric model and a graphical algorithm for a sequencing problem. *Operations Research*, 11(6):889–900, 1963.

[5] D.-H. Lee, Z. Cao, and Q. Meng. Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. *International Journal of Production Economics*, 107:115–124, 2007.

[6] W. C. Ng. Crane scheduling in container yardswith inter-crane interference. *European Journal of Operational Research*, 164:64–78, 2005.

[7] W. C. Ng and K. L. Mak. An effective heuristic for scheduling a yard crane to handle jobs with different ready times. *Engineering Optimization*, 37:867–877, 2005.

[8] R. Stahlbock and S. Voß. Operations research at container terminals: a literature update. *OR Spectrum*, 30:1–52, 2008.

[9] D. Steenken, S. Voß, and R. Stahlbock. Container terminal operations and operations research – a classification and literature review. *OR Spectrum*, 26:3–49, 2004.

[10] I. F. A. Vis. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research*, 170:677–709, 2006.

[11] I. F. A. Vis and H. J. Carlo. Sequencing two cooperating automated stacking cranes in a container terminal. *Transportation Science*, 44:169–182, 2010.